Lab 02
ROS Basics

Wan Fang

Visiting Scholar

SUSTech Institute of Robotics

sophie.fwan@hotmail.com

Agenda

Week 02, Friday, March 1

- ROS Basics
  - ROS Architecture
  - ROS File System
  - ROS Computation Graph & Communication
  - ROS tools
  - Homework

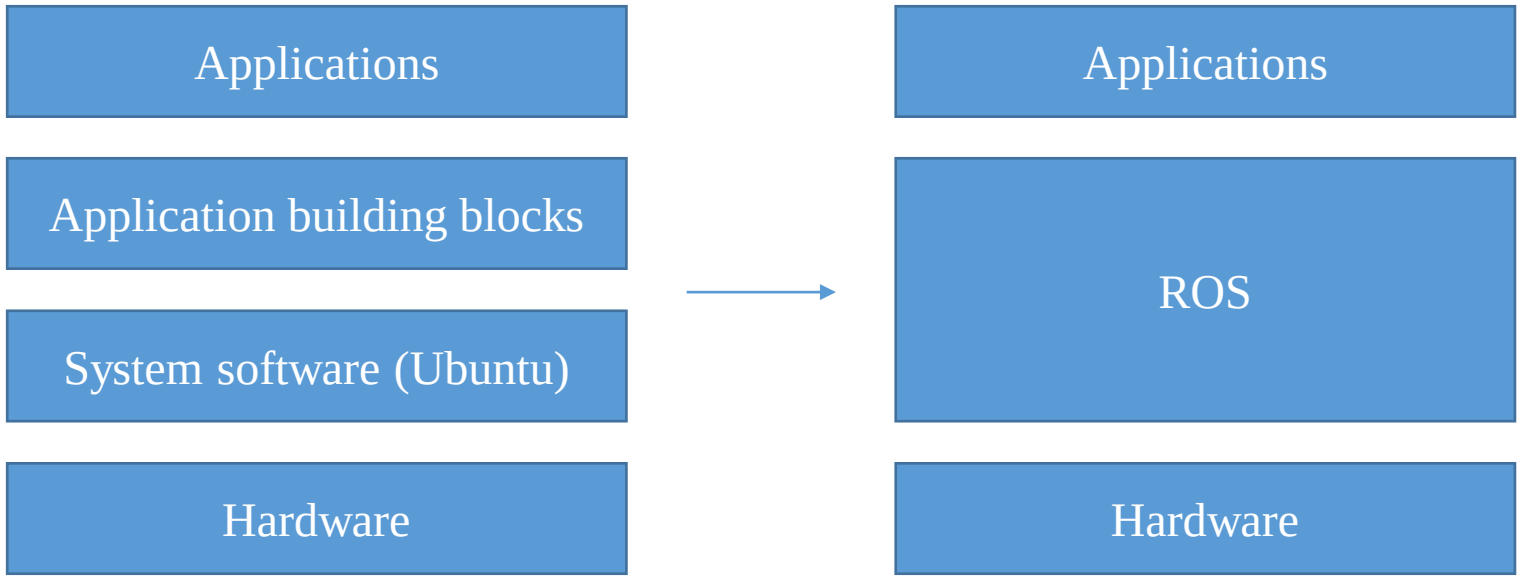
AncoraSIR.com

SUSTech
Southern University of Science and Technology

ROS Architecture

Comparison: the robotics ecosystem

- ROS is a middle layer



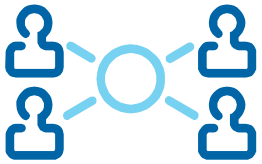
AncoraSIR.com

ROS Architecture

Conceptual Levels of Design

- **ROS Community:**

  ROS Distributions, Repositories



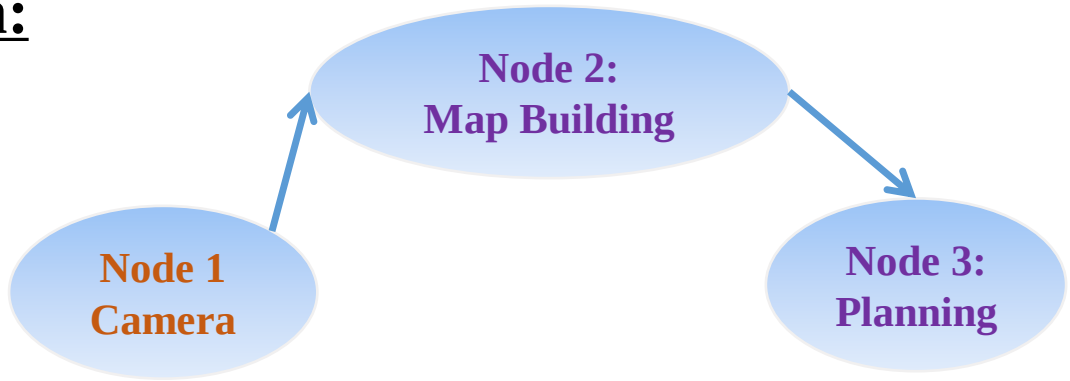
- **File-system level:**

  ROS Tools for managing source code, build instructions, and message definitions.

- **Computation Graph:**

  Peer-to-Peer Network of

  ROS nodes (processes).

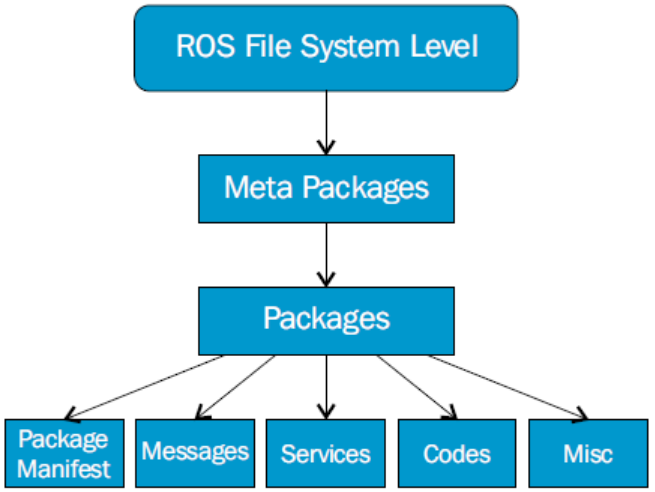


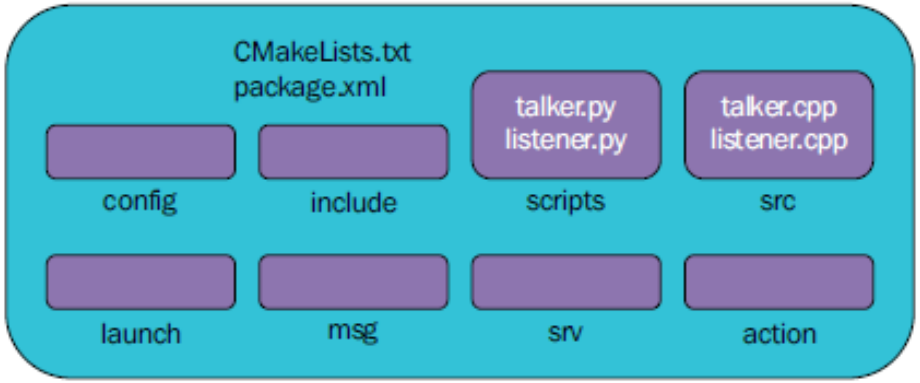
AncoraSIR.com

ROS File System

ROS Packages

- ROS software is organized into *packages*.
  - ***Package Manifests***: Manifests (package.xml) provide metadata about a package, including its name, version, description, license information, dependencies.
  - ***Message (msg)***: Message descriptions, define the data structures for messages sent in ROS.
  - ***Service (srv)***: Service descriptions, define the request and response data structures for services in ROS.
  - ***Codes***: C++/Python source codes.





AncoraSIR.com

ROS File System

Catkin Build Tools and ROS workspace

- ***catkin*** is the ROS build system to generate executables, libraries, and interfaces.
  - We suggest to use the ***Catkin Command Line Tools***
  - Installation: $sudo apt-get install python-catkin-tools
- A ROS Workspace is simply a set of directories in which a related set of ROS code lives.
  - Create a ROS workspace: $mkdir –p ~/catkin_ws/src
  - Add a ROS package under /catkin_ws/src: $git clone
  - Build a ROS package: $cd.. $catkin build
  - Set the environment variabe: $source ~/catkin_ws/devel/setup.bash

AncoraSIR.com

ROS File System

Catkin Build Tools

The catkin workspace contains the following spaces

Work here

src

The *source space* contains the source code. This is where you can clone, create, and edit source code for the packages you want to build.

Don't touch

build

The *build space* is where CMake is invoked to build the packages in the source space. Cache information and other intermediate files are kept here.

Don't touch

devel

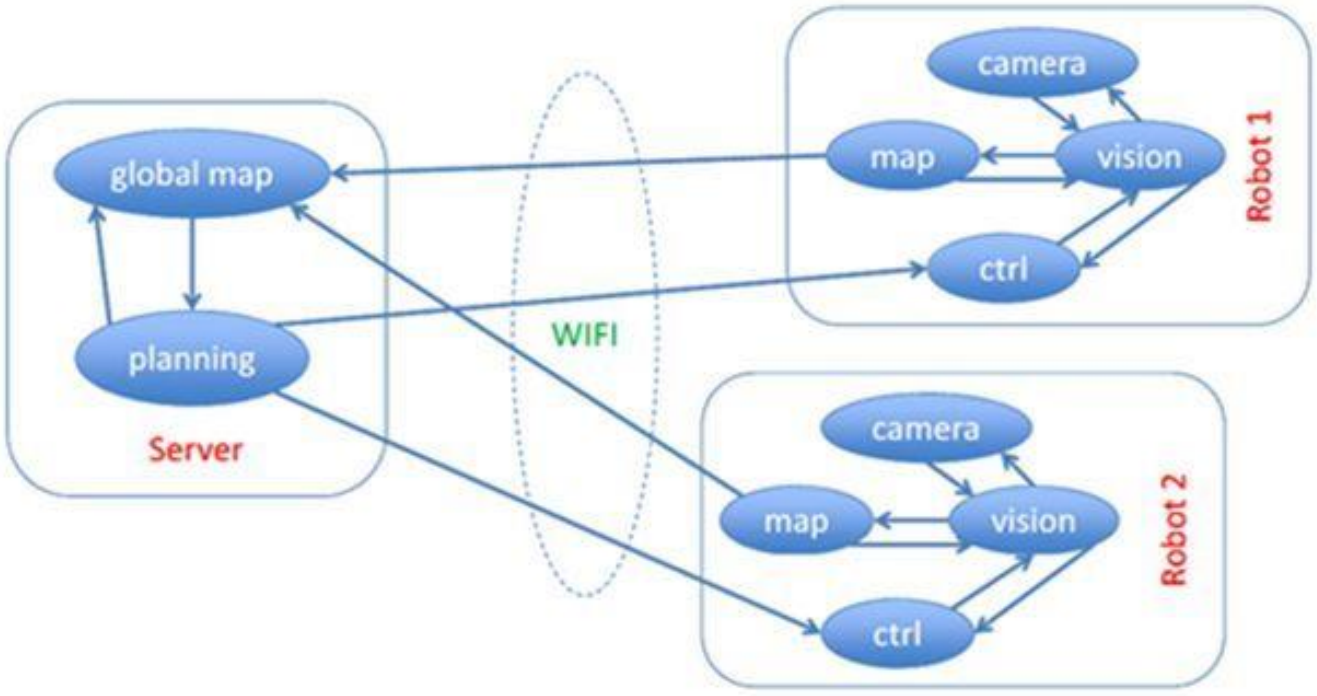
The *development (devel) space* is where built targets are placed (prior to being installed).

AncoraSIR.com

ROS Computation Graph

Overview

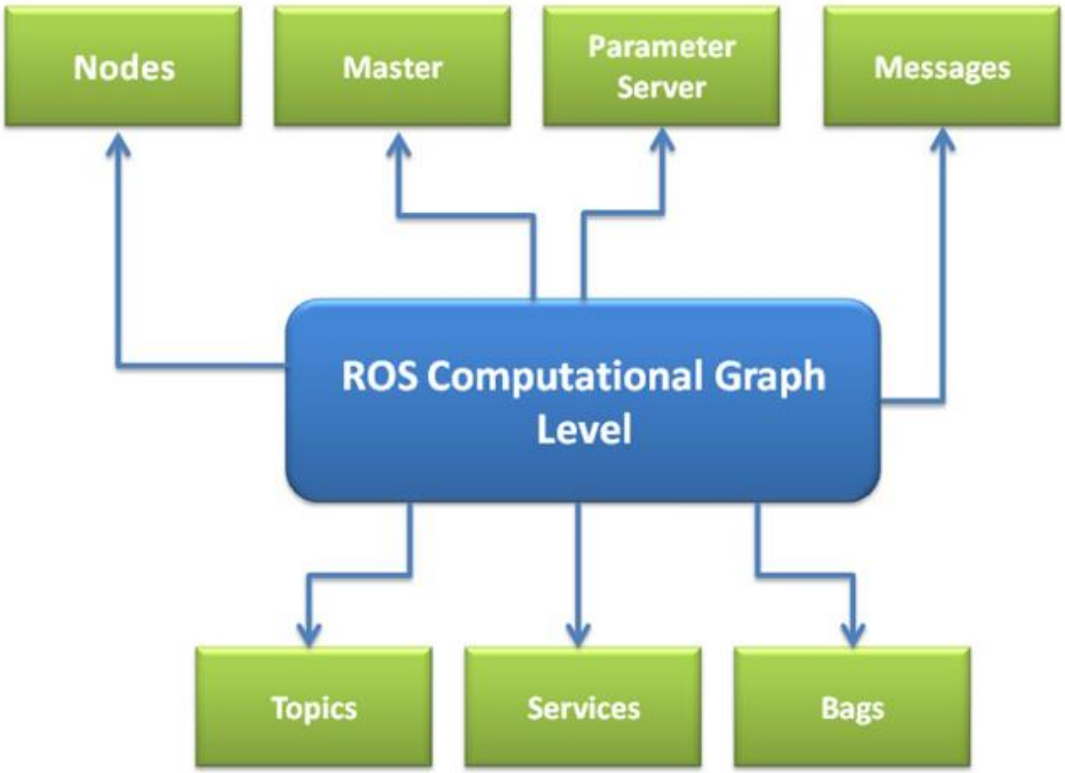
- The ***Computation Graph*** is the peer-to-peer network of ROS processes that are processing data together.



AncoraSIR.com

ROS Computation Graph

- The basic Computation Graph concepts of ROS are ***nodes***, ***master***, ***parameter server***, ***messages***, ***services***, ***topics*** and ***bags***.

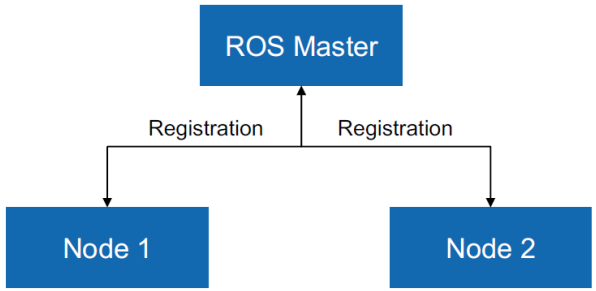


AncoraSIR.com

ROS Computation Graph

ROS Master & Node

- ROS ***Master*** manages the communication between nodes
- ROS ***node*** is a single-purpose, executable program individually compiled, executed, managed and organized in ***packages***
- Every node registers at startup with the master
- Start a master: $roscore
- Run a node: $rosrun package_name node_name

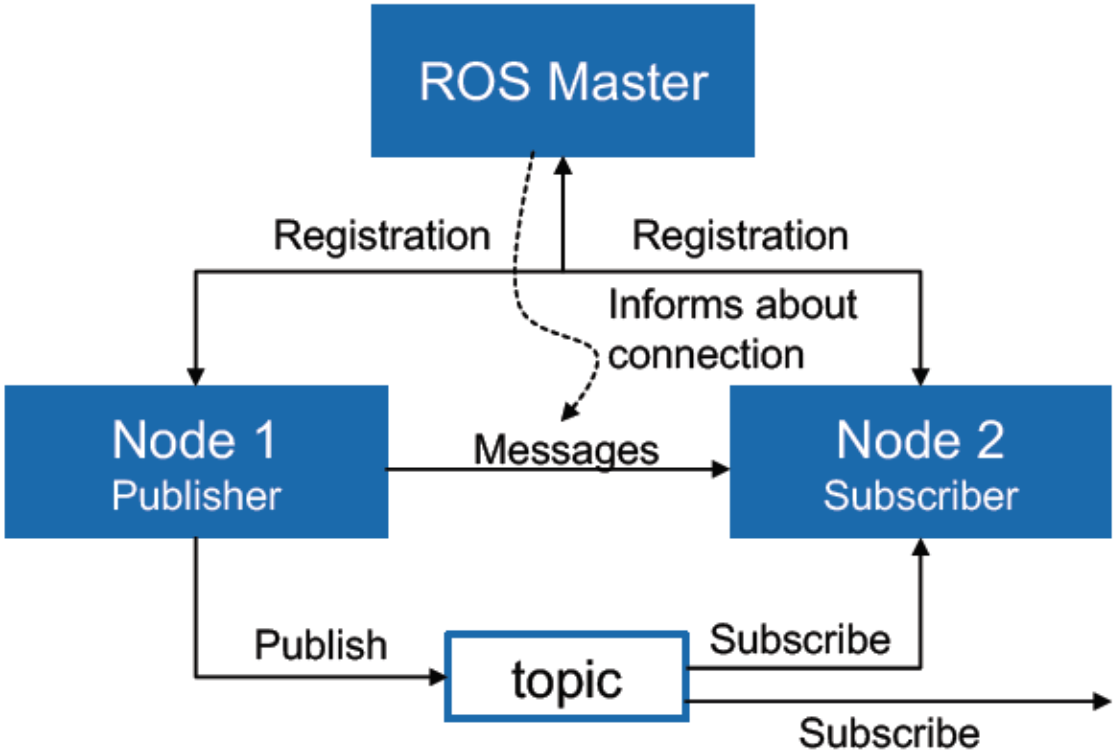


AncoraSIR.com

ROS Computation Graph

ROS Topic

- ROS ***topics*** are named buses in which ROS nodes exchange messages. Topics can publish and subscribe.

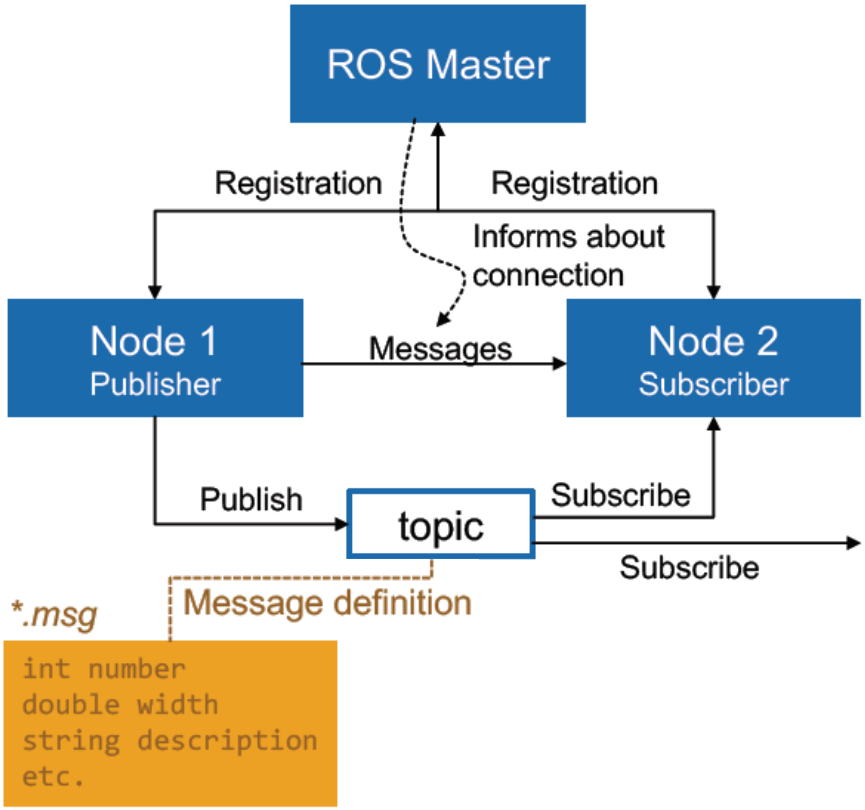


AncoraSIR.com

ROS Computation Graph

ROS Message

- ROS ***message*** is data structure defining the type of a topic
- Compromised of a nested structure of
  - integers
  - floats
  - booleans
  - strings
  - arrays of objects
- Defined in *.msg files



AncoraSIR.com

ROS Message

Some typical message defined in ROS

geometry_msgs/Point.msg

```
float64 x
float64 y
float64 z
```

sensor_msgs/Image.msg

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
 uint32 seq
 time stamp
 string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

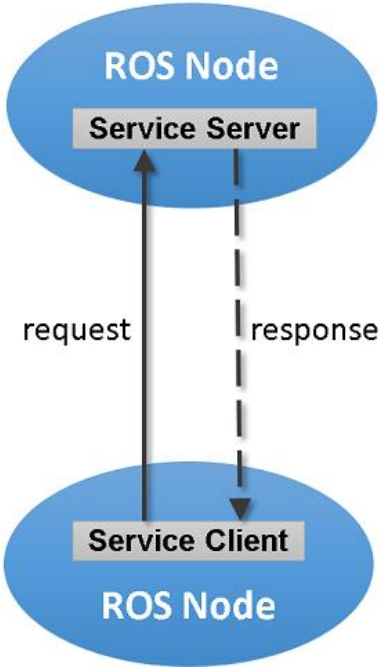
AncoraSIR.com

SUSTech
Southern University of Science and Technology

ROS Computation Graph

ROS Service

- The ROS ***services*** are a type request/response communication between ROS nodes.
- An example service description format is as follows:



PickPlace.srv

```
# request
std_msgs/String object_name
geometry_msgs/Pose pick_pose
---
# response
bool success
```

BionicDL-CobotLearning-Project1_PickPlace.srv.html

AncoraSIR.com

ROS tools

roslaunch

- ***launch*** is a tool for launching multiple nodes (as well as setting parameters)
- Are written in XML as *.launch files
- Start a launch file from a package:

  $roslaunch package_name file_name.launch

! Attention when copy & pasting code from the internet

talker_listener.launch

```
<launch>
    <node name="listener" pkg="roscpp_tutorials" type="listener" output="screen"/>
    <node name="talker" pkg="roscpp_tutorials" type="talker" output="screen"/>
</launch>
```

! Notice the syntax difference for self-closing tags: `<tag></tag>` and `<tag/>`

- **launch**: Root element of the launch file
- **node**: Each *<node>* tag specifies a node to be launched
- **name**: Name of the node (free to choose)
- **pkg**: Package containing the node
- **type**: Type of the node, there must be a corresponding executable with the same name
- **output**: Specifies where to output log messages (`screen`: console, `log`: log file)

ROS tools

rosparam

- Nodes use the ***parameter server*** to store and retrieve parameters at runtime.
- The ***rosparam*** tool enables command-line setting and getting of parameters as well as loading and dumping ***parameter server*** state to a file.
  - set parameter: $ rosparam set parameter_name value
  - get parameter: $ rosparam get parameter_name
  - load parameters from file: $ rosparam load config.yaml
  - list parameter names: $ rosparam list

config.yaml

```
camera:
  left:
    name: left_camera
    exposure: 1
  right:
    name: right_camera
    exposure: 1.1
```

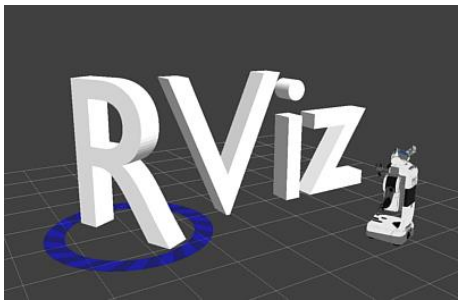
AncoraSIR.com

SUSTech
Southern University of Science and Technology

ROS tools

Rviz

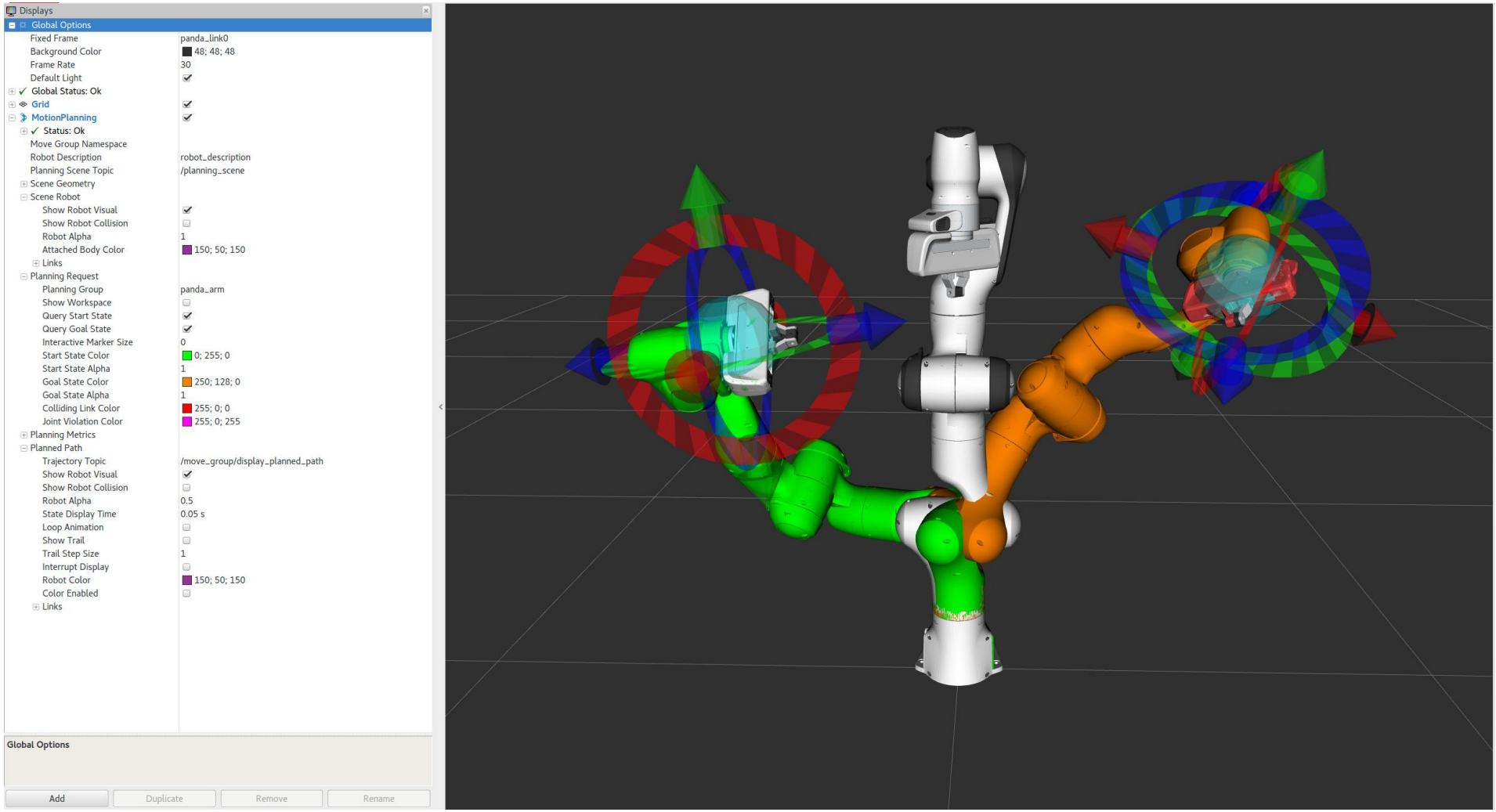
- ***RViz*** (or rviz) stands for ROS Visualization tool
  - Visualize any type of ***sensor data*** being published over a ROS topic like camera images, point clouds, ultrasonic measurements, Lidar data, inertial measurements, etc.
  - Visualize live ***joint angle values*** from a robot and hence construct a real-time 3D representation of any robot.
  - Interactive tools to publish user information
  - Save and load setup as RViz configuration



AncoraSIR.com

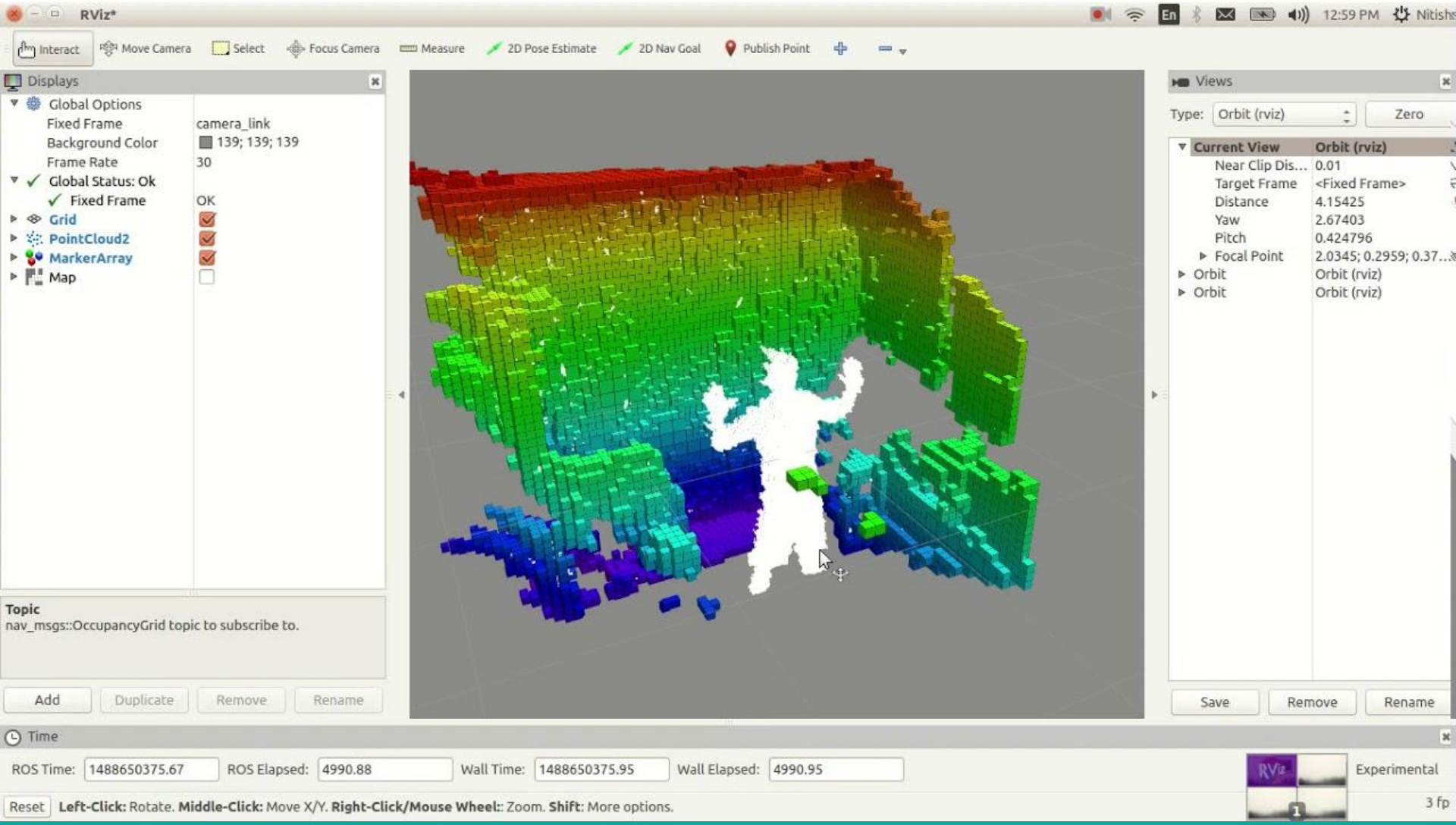
RViz

Visualize live joint angle values



RViz

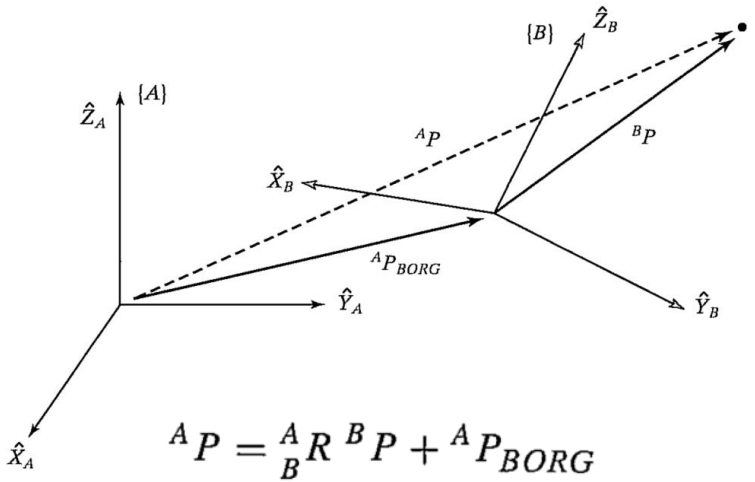
Visualize sensor data

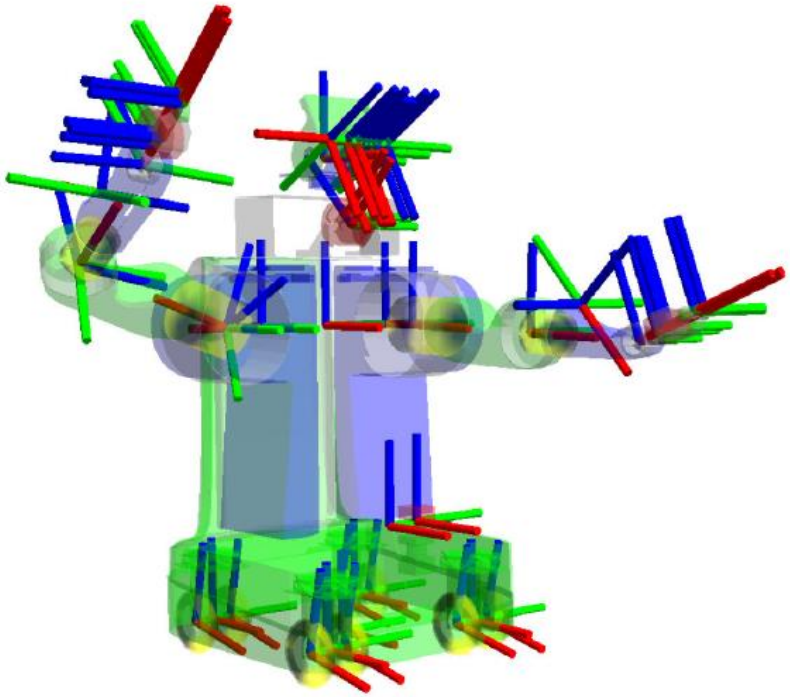


ROS tools

tf

- ***tf*** is a ros tool for coordinate transformation
- Robot has many frames and coordinates, which are moving
- Where is the gripper relative to the arm?



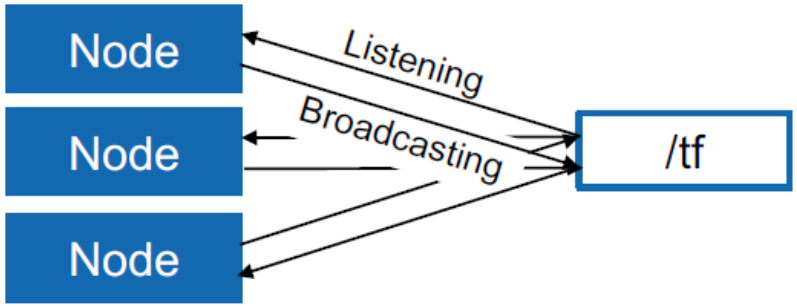


AncoraSIR.com

tf

What does it do?

- Tool for ***keeping track*** of coordinate frames over time.
- ***Maintains*** relationship between coordinate frames in a tree structure buffered in time.
- Lets the user ***transform*** points, vectors, etc. between coordinate frames at desired time.
- Implemented as publisher/subscriber model on the topics /tf and /tf_static.



AncoraSIR.com

tf

How to use tf tools?

Command line

Print information about the current tranform tree

```
> rosrun tf tf_monitor
```

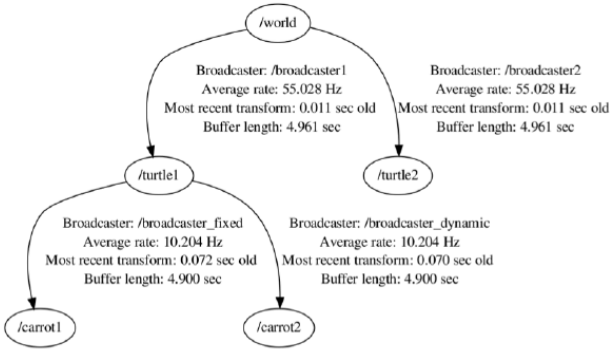
Print information about the transform between two frames

```
> rosrun tf tf_echo
    source_frame target_frame
```

View Frames

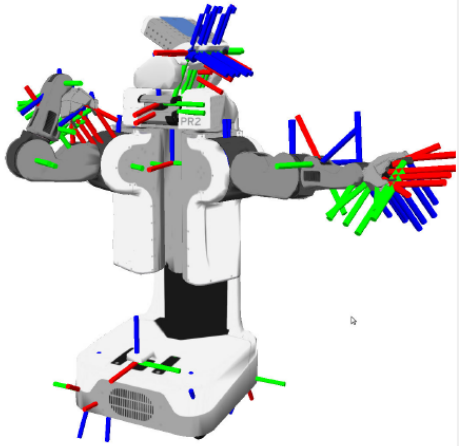
Creates a visual graph (PDF) of the transform tree

```
> rosrun tf view_frames
```



RViz

3D visualization of the transforms



AncoraSIR.com

Homework

- Review the concepts covered in this lab session.
- Get familiar with the structure and codes for Project1: Simulate A Picking Robot in Gazebo
  - Codes and instructions can be found at
  - https://github.com/ancorasir/BionicDL-CobotLearning-Project1
  - Project 1 due in week 4 (March 17).

sophie.fwan@hotmail.com

AncoraSIR.com

Thank you!

Prof. Song Chaoyang

- Dr. Wan Fang (sophie.fwan@hotmail.com)

AncoraSIR.com